Compressor

CCSDS 123 Multispectral & Hyperspectral Image Compression upcoming standard

Luca Fossati (Luca.Fossati@esa.int) – 12/04/2012

The software implements the lossless compression for hyperspectral and multispectral images according to the Draft Recommended Standard CCSDS 123.0-R-1 as of 09/11/2011. Both the compressor and decompressor are implemented.

It runs natively under Unix and Microsoft Windows without the need to have external libraries and other dependences.

Licensing

The Software is distributed under the "European Space Agency Public License – v2.0", whose full text it delivered together with the software source code.

All Distribution of the Software and/or Modifications, as Source Code or Object Code, must be, as a whole, under the terms of the European Space Agency Public License – v2.0.

If You Distribute the Software and/or Modifications as Object Code, You must:

(a) provide in addition a copy of the Source Code of the Software and/or Modifications to each recipient; or

(b) make the Source Code of the Software and/or Modifications freely accessible by reasonable

means for anyone who possesses the Object Code or received the Software and/or Modifications from You, and inform recipients how to obtain a copy of the Source Code.

The Software is provided to You on an “as is” basis and without warranties of any kind, including without limitation merchantability, fitness for a particular purpose, absence of defects or errors, accuracy or non-infringement of intellectual property rights.

Except as expressly set forth in the "European Space Agency Public License – v2.0", neither Licensor nor any Contributor shall be liable, including, without limitation, for direct, indirect, incidental, or consequential damages (including without limitation loss of profit), however caused and on any theory of liability, arising in any way out of the use or Distribution of the Software or the exercise of any rights under this License, even if You have been advised of the possibility of such damages.

Compressor - Command Line options:

Generic

* *input:* path to the file containing the residuals produced by the predictor
* *output:* path to the file that will be produced by the encoder
* *rows:* number of rows in each band of the input image (called Ny in the standard)
* *columns:* number of columns in each band of the input image (called Nx in the standard)
* *bands:* number of bands in the input image (called Nz in the standard)
* *in\_format:* format (BSQ or BI) with which the samples are arranged in the residuals file
* *in\_depth:* considered just for the input BI format, interleaving depth (M)
* *in\_byte\_ordering:* LITTLE|BIG: byte ordering for the input uncompressed file; this option is considered only if also the *reg\_input* option is specified.
* *out\_format:* format (BSQ or BI) with which the samples shall be arranged in the output file
* *out\_depth:* considered just for the output BI format, interleaving depth
* *dyn\_range:* dynamic range of each of the samples in the image.
* *word\_len:*B, number of bytes composing an output word.
* *reg\_input:* specifies that the input file containing the original image has each sample coded with 16bits, even though the dynamic range can be smaller than that. If this option is not specified, the compressor assumes that exactly *dyn\_range* bits are used to code each of the uncompressed samples.

Predictor

* *signed\_sample:* if specified, means that the samples of the input image are signed integer numbers
* *pred\_bands:* the number of bands used for the prediction
* *full:* if specified, full prediction mode shall be used
* *neighbour\_sum:* if specified, neighbour oriented sum will be used (as opposed to column oriented)
* *reg\_size:* register size R
* *w\_resolution:* weight component resolution Omega
* *w\_interval:* weight update scaling exponent change interval t\_inc
* *w\_initial:* weight update scaling exponent initial parameter v\_min
* *w\_final:* weight update scaling exponent final parameter v\_max
* *w\_init\_resolution:* weight initialization resolution Q
* *weight\_init\_file:* file containing the weight initialization table; if not specified, the default weight initialization shall be used.

Encoder

* *sample\_adaptive:* if specified the sample adaptive encoder shall be used as opposed to the block adaptive.
* *u\_max:* for the sample adaptive encoder, unary length limit (Umax)
* *y\_star*: for the sample adaptive encoder, Rescaling Counter Size (gamma\*)
* *y\_0:* for the sample adaptive encoder, Initial Count Exponent (gamma\_0)
* *k:* for the sample adaptive encoder, Accumulator Initialization constant
* *k\_init\_file:* for the sample adaptive encoder, File containing the accumulator initialization table (cannot be specified together with the Accumulator Initialization constant: only one of the two can be used)
* *block\_size:* for the block adaptive encoder, the block size J
* *restricted\_enc:* for the block adaptive encoder, if specified and the dynamic range D is <= 4, the restricted mode will be used
* *ref\_interval:* for the block adaptive encoder, the reference sample interval r

The accumulator initialization table and the weight initialization table simply consists of a text file with a number on each line; the weights initialization table has an empty line every Cz values (i.e. every time we move to the successive band). Also note that the weight initialization table has a constant number of values for each band, even though, of course, they are not used for the first bands; the values assigned to these non-used weights are non-important.

Decompressor - Command Line options:

* *input:* the path to the compressed file
* *output:* the path to the uncompressed file to be created
* *out\_format:* format (BSQ or BI) with which the samples shall be arranged in the output file
* *out\_depth:* considered just for the output BI format, interleaving depth
* *out\_byte\_ordering:* LITTLE|BIG: byte ordering for the output uncompressed file

Testing

The following command line options are used to cross-check the results produced by NASA on the artificial Test Pattern Images provided:

* sample-adaptive coding, output word size B=8 bytes, band interleaving with interleave depth M=7, register size R=32, weight resolution Omega=14, compressed image file "testptn\_O14R32\_gpo2.fl".

*./compressor --input testptn\_w100h36b17\_16u.bip --output testptn\_O14R32\_gpo2\_esa.fl --rows 36 --columns 100 --bands 17 --in\_format BI --in\_depth 17 --dyn\_range 16 --word\_len 8 --out\_format BI --out\_depth 7 --sample\_adaptive --u\_max 18 --y\_star 6 --y\_0 1 --k 7 --pred\_bands 15 --full --reg\_size 32 --w\_resolution 14 --w\_interval 32 --w\_initial -6 --w\_final -6*

* sample-adaptive coding, output word size B=7 bytes, band interleaving with interleave depth M=5, , register size R=37, weight resolution Omega=19, compressed image file "testptn\_O19R37\_gpo2.fl".

*./compressor --input testptn\_w100h36b17\_16u.bip --output testptn\_O19R37\_gpo2\_esa.fl --rows 36 --columns 100 --bands 17 --in\_format BI --in\_depth 17 --dyn\_range 16 --word\_len 7 --out\_format BI --out\_depth 5 --sample\_adaptive --u\_max 18 --y\_star 6 --y\_0 1 --k 7 --pred\_bands 15 --full --reg\_size 37 --w\_resolution 19 --w\_interval 32 --w\_initial -6 --w\_final -6*

* block-adaptive coding, output word size B=7 bytes, BSQ, register size R=32, weight resolution Omega=14, compressed image file "testptn\_O14R32\_rice.fl".

*./compressor --input testptn\_w100h36b17\_16u.bip --output testptn\_O14R32\_rice\_esa.fl --rows 36 --columns 100 --bands 17 --in\_format BI --in\_depth 17 --dyn\_range 16 --word\_len 7 --out\_format BSQ --block\_size 16 --ref\_interval 256 --pred\_bands 15 --full --reg\_size 32 --w\_resolution 14 --w\_interval 32 --w\_initial -6 --w\_final -6*

A Python script (*run\_tests.py*) is provided for running the compressor over the CCSDS reference of images and for automatically verifying the result of the compression; in order to be able to use the script, the following input has to be provided:

* CSV (Comma Separated Values) file corresponding to the information contained in the Excel file YellowBook\_123\_table-2012Jan4 (i.e. the parameters determining the behaviour of the compressor). At the end of each line of the file, two additional columns were added: the first to indicate if the input image is in BIG (B) or LITTLE (L) endian format, the second to indicate the band ordering in the input image (BIP or BSQ); for the CCSDS reference set of images, such CSV file is provided together with the compressor software.
* The folder containing the original uncompressed images.
* The folder containing already compressed images (used for verification of the compression).
* The folder containing already computed mapped residuals (used for verification of the predictor); this is useful to determine, in case of an error, if the error lies in the predictor or in the entropy encoder.

The images in the three folders have to be named *Instrument\_Image*, where Instrument and Image are the first two columns of the previously mentioned Excel file. The case or the extension of the images’ file names is not important.

The files containing the accumulator initialization tables and the weight initialization tables (where necessary) have to be present in the folder containing the compressed images and they have to be in the format mentioned above.

Note also that the script only runs under Unix and it requires the presence of the *diff* program to check the bit-wise equality between the result of the compressor and the reference compressed images and residual files.

Code Structure

The compressor is written using the C language and it does not uses any external library. The command line parsing uses the *getopt()* routine as specified in the POSIX.2 and POSIX.1-2001 standards; for native compilation under Microsoft Windows, the implementation of such routine is provided in separate files. As such, under Microsoft Windows, this product includes software developed by the University of California, Berkeley and its contributors.

The compressor is composed of 4 files:

1. console\_main.c

*Contains the main routine; it mostly takes care of parsing the command line options and starting the compression by calling the predict and encode routines. Command line parsing is based on the getopt library, provided by the OS when using UNIX based systems, otherwise sources are shipped together with the compressor for native compilation under Microsoft Windows.*

1. predictor.c

*As the name suggests, this file takes care of performing the prediction of the image samples and, using the result, computing the residual values. Depending on whether macro NO\_COMPUTE\_LOCAL is defined or not in file predictor.h, local differences are either pre-computed and stored in a local buffer or computed every time they are needed. When pre-computed, the local differences will use a large memory portion, and prediction will fail if your system has a limited amount of memory; on the other hand, compression will be faster, as it avoids computing the same values over and over.*

*The predict routine takes in input the name of the files containing the samples and it produces an array of unsigned short integers (each 16 bits wide) representing the mapped residuals; such residuals are stored in the array in BSQ order.*

1. entropy\_encoder.c

*This file contains all the routines for the encoding of the residuals into the final compressed image; it also produces the header appended to the compressed file. As described in the standard, two different compression mechanisms might be chosen: sample adaptive and Rice adaptive coding (according to CCSDS 121 standard).*

*The encode routine takes in input the residuals array (where residuals are represented by 16 bit wide unsigned integers and stored in BSQ order) and the name of the output file where to store the compressed image.*

1. utils.c

*Finally such file contains routines used all over the prediction and encoding processes but not strictly related to them. Examples of such routines are: conversion among BSQ, BI, and BIP, parsing of the input image’s samples, storing on file of the bits composing the output image, determining the machine’s endianes, etc.*